

D0Note 3957: Missing ET Reconstruction: Variables and Methods

Lee Sawyer
sawyer@phys.latech.edu

Alan Stone
alstone@fnal.gov

*Louisiana Tech University
Ruston, La 71272*

July 1, 2003

Abstract

The missing E_T reconstruction package `missingET` is described. A full listing of the variables calculated by the package, and the methods for accessing these variables, are presented. A history of the variables available in previous production releases are included in an appendix.

1 Overview

Calculation of the total scalar and missing transverse energy in an event is an important tool for physics at the Tevatron. The calculation of missing transverse energy, or \cancel{E}_T , combines quantities from central tracking (for primary vertex determination), calorimeter (for energy deposition) and muon detector (for the muon correction to the energy in the event), and is clearly a first pass at determining the energy flow in the event.

We define the x - and y -components of the visible energy in the calorimeter as

$$E_{x,y}^{\text{vis}} = \sum_{\text{cells}} E_i^{x,y}$$

. Then the corresponding projections of the calorimeter missing energy are $\cancel{E}_x = -E_x^{\text{vis}}$ and $\cancel{E}_y = -E_y^{\text{vis}}$. The missing transverse energy in the calorimeter is then $\cancel{E}_T = \sqrt{(\cancel{E}_x)^2 + (\cancel{E}_y)^2}$.

The scalar E_T in the calorimeter is defined as

$$\sum_{\text{cells}} \sqrt{(E_i^x)^2 + (E_i^y)^2}$$

Similar calculations can be made based on towers rather than cells [1]. The cell and tower values are returned from the `CalDataChunk` using the vertex information obtained from `VertexInfo::get_Vertex`.

The calorimeter \cancel{E}_T and scalar E_T values are corrected for the presence of reconstructed muons in the event. The contribution of muon momentum to the total momentum of the final state is calculated by summing over tracks in the `MuonParticleChunk` passing the "tight" set of quality cuts. Then the muon-corrected \cancel{E}_T is given by

$$\cancel{E}_T^{\text{CAL+MUON}} = \sqrt{(\cancel{E}_x - p_x^{\text{muons}})^2 + (\cancel{E}_y - p_y^{\text{muons}})^2}$$

, where $p_{x,y}^{\text{muons}}$ are the summed x - and y -components of the tight muon momenta.

Specialized versions of the visible E_T , missing E_T , and scalar E_T calorimetric variables are calculated by layer (*e.g.* EM, FH1, ICD, etc.), with a cell or tower energy threshold, or for a limited region of eta coverage. Thresholds and eta limits are set in the RCP file `missingET\rcp\MissingET.rcp`. A complete list of variables calculated by the `missingET` package is given below.

In addition to the missing E_T and scalar E_T quantities, "rings" in bins of detector eta are calculated with respect to a $z = 0$ primary vertex. These rings are stored by x - and y -component for the electromagnetic and hadronic summed E_T in each of the 74 η -bins. A more complete description of the rings and their use in recalculating missing E_T for different event vertices is given in ref. [2].

All variables calculated by the `missingET` package are available for other reconstruction programs via accessor "get" and "print" accessor methods (described below). The post-reconstruction analysis package `met_analyze` produces ROOT tuple blocks containing these variables.

In the thumbnail, three sets of variables are currently stored:

- \cancel{E}_T , \cancel{E}_x , \cancel{E}_y , and scalar E_T with no E_T or η limits applied.
- \cancel{E}_T , \cancel{E}_x , \cancel{E}_y , and scalar E_T with E_T threshold only.
- \cancel{E}_T , \cancel{E}_x , \cancel{E}_y , and scalar E_T with E_T and η limits.
- The ring variables.
- The z -coordinate of the primary vertex used to calculate the missing E_T variables.

2 Variables and Methods

The variables calculated by the `missingET` package have the following naming convention:

- SET: scalar ET obtained as the sum of cell energy*abs(sin(theta)). A cell with negative energy will give a NEGATIVE contribution to SET.
- VETx: x -component of vector E_T obtained as the sum of the cell energy*sin(th)*cos(ph)
- VETy: y -component of vector E_T obtained as the sum of the cell energy*sin(th)*sin(ph)
- The x and y component of the Missing ET (METx, METy) are simply:
 $\text{METx}=-\text{VETx}$
 $\text{METy}=-\text{VETy}$
- $\text{MET}=\sqrt{\text{METx}^2+\text{METy}^2}$, and of course $\text{MET}=\text{VET}$.

In the detailed description of the variables, we use VETx and VETy, in order to have a symmetric (same sign) treatment of calorimeter and muons, e.g. VETx=+VETCALOx+VETMUONx, etc...

For production release p14.03.01 and later, the following variables are calculated (accessor methods are given with each set of variables):

MET for CAL + ICD Towers

```
float _SETT;      void getMETT(float &SETT,
float _METTx;      float &METTx, float &METTy, float &METT);
float _METTy;
float _METT;
```

MET for CAL + ICD Towers + Muon correction

```
float _SETTM;      void getMETTM(float &SETTM,
float _METTMx;      float &METTMx, float &METTMy, float &METTM);
float _METTMy;
float _METTM;
```

MET for CAL + ICD Cells

```
float _SETC;      void getMETC(float &SETC,
float _METCx;      float &METCx, float &METCy, float &METC);
float _METCy;      void getVETC(float &SETC,
float _METC;      float &VETCx, float &VETCy, float &METC);
float _VETCx;
float _VETCy;
```

MET for CAL + ICD Cells + Muon correction

```
float _SETCM;      void getMETCM(float &SETCM,
float _METCMx;      float &METCMx, float &METCMy, float &METCM);
float _METCMy;     void getVETCM(float &SETCM,
float _METCM;      float &VETCMx, float &VETCMy, float &METCM);
float _VETCMx;
float _VETCMy;
```

MET for CAL + ICD Cells layers 1-14 (no CH)

```
float _SETD;      void getMETD(float &SETD,
float _VETDx;      float &METDx, float &METDy, float &METD);
float _VETDy;     void getVETD(float &SETD,
float _METD;      float &VETDx, float &VETDy, float &METD);
```

MET for CAL + ICD Cells layers 1-14 (no CH) + Muon correction

```
float _SETDM;      void getMETDM(float &SETDM,
float _VETDMx;      float &METDMx, float &METDMy, float &METDM);
float _VETDMy;     void getVETDM(float &SETDM,
float _METDM;      float &VETDMx, float &VETDMy, float &METDM);
```

MET for EM layers (1-7)

```
float _SETEM;      void getVETEM(float &SETEM,
float _VETEMx;      float &VETEMx, float &VETEMy, float &METEM);
float _VETEMY;
float _METEM;
```

MET for Massless Gap layers (8&10)

```
float _SETMG;      void getVETMG(float &SETMG,
float _VETMGx;      float &VETMGx, float &VETMGy, float &METMG);
float _VETMGy;
float _METMG;
```

MET for FH layers (11-14)

```
float _SETFH;      void getVETFH(float &SETFH,
float _VETFHx;      float &VETFHx, float &VETFHy, float &METFH);
float _VETFHy;
float _METFH;
```

MET for CH layers (15-17)

```
float _SETCH;      void getVETCH(float &SETCH,
float _VETCHx;      float &VETCHx, float &VETCHy, float &METCH);
float _VETCHy;
float _METCH;
```

MET for CAL + ICD Cells below "edge" (above eta limit) (no threshold)

```
float _EDge;      void getVETED(float &EDGE, float &SETED,
float _SETED;          float &VETEDx, float &VETEDy, float &METED);
float _VETEDx;
float _VETEDy;
float _METED;
```

Vis. Et for ICD Cells

```
float _SETICD;      void getVETICD(float &SETICD,
float _VETICDx;          float &VETICDx, float &VETICDy, float &VETICD);
float _VETICDy;
float _VETICD;
float _METICD;
```

Vis. Et for Isolated "Hot" Cells found by NADA algorithm.

```
float _SETNADA;      void getVETNADA(float &SETNADA,
float _VETNADAx;          float &VETNADAx, float &VETNADAY, float &VETNADA);
float _VETNADAY;
float _VETNADA;
float _METNADA;
```

Tight Muons

```
float _VETMUONx;  void getVETMUON(float &VETMUONx,
float _VETMUONY;          float &VETMUONY, float &VETMUONz, float &VETMUON);
float _VETMUONz;
float _VETMUON;
float _METMUON;
```

MET for negative cells

```
float _SETNG;      void getVETNG(float &SETNG,
float _VETNGx;          float &VETNGx, float &VETNGy, float &METNG);
float _VETNGy;
float _METNG;
```

MET for noise cells, based on the CalT42 algorithm [4].

```
float _SET42;      void getVET42(float &SET42,
float _VET42x;          float &VET42x, float &VET42y, float &MET42);
float _VET42y;
float _MET42;
```

In addition, the rings variables are accessed by the method

```
void getRings(const float *&RingEMx, const float *&RingEMy,
              const float *&RingHDx, const float *&RingHDy);
```

while the z -coordinate of the primary vertex used to calculate the missing E_T can be accessed with the method

```
void getZvertex(float &Zvertex);
```

The "visible" momentum of tight muons used to correct the missing E_T is obtained via

```
void getVisMu(float &visMuX, float &visMuY, float &visMuPt);}
```

3 Acknowledgements

The original version of the `missingET` package was coded by John Womersley in 1998. Joe Steele was responsible for the MET rings revertexting, which is described in detail in D0Note 3895[2]. Laurent Duflot provided assistance with `met_analyze`, which was originally part of `jet_analyze`. Serban Protopopescu and Stephanie Baffioni provided part of the thumbnail work. Sophie Trincaz-Duvoud found some bug fixes and provided the NADA interface[3]. Gregorio Bernardi, Vishnu Zutshi, and Bob Hirosky had input into the definition of variables. Harry Melanson and Geoff Savage were software consultants. Scott Snyder provided fixes to the LIBRARIES and OBJECTS and LIBDEPS files. Markus Klute, Marco Verzocchi, and Tobi Golling (among others) provided feedback on variables and algorithms.

4 Appendix A: Variables by Production Release

As of the release p14.03.01, the philosophy is to calculate the contribution to the missing E_T by layer of the calorimeter. Previously, in p11 through p13, the philosophy was to calculate the missing E_T with different E_T and η thresholds, plus a few specialized calculation such as the ICD/MG contribution, NADA cells, etc. Prior to p11, only a small set of tower-based variables were calculated.

For events reconstructed with release prior to 11 (these variables are still retained for backwards compatibility) the following tower-based quantities were calculated:

```
///////////////////////////////
//          old variable set          //
// no Tower Energy or Eta Limits
float _MET;
float _MEx;
float _MEy;
float _ScalarET;

// with Tower Energy limit only
float _MET_noeta;
float _MEx_noeta;
```

```

float _MEy_noeta;
float _ScalarET_noeta;

// with Tower Energy and Eta limits
float _MET_weta;
float _MEx_weta;
float _MEy_weta;
float _ScalarET_weta;

// Ring variables: 42 ieta rings each with EM & HAD components
float _RingEMx[NRING];
float _RingEMy[NRING];
float _RingHdx[NRING];
float _RingHDy[NRING];

float _visMuMom[4];
float _Zvertex;

```

The corresponding accessor methods for these variables are

```

float getScalarET();
void getMExy(float &ETx, float &ETy);
float getMET();
float getScalarET_noeta();
void getMExy_noeta(float &ETx_noeta, float &ETy_noeta);
float getMET_noeta();
float getScalarET_weta();
void getMExy_weta(float &ETx_weta, float &ETy_weta);
float getMET_weta();

```

Nota Bene ! However, as of p14.03.01, the methods `getScalarET`, `getMExy`, and `getMET` return the METC (cell-level, all layers) values, since many users were interpreting these accessors as the "default" accessor methods.

For release p11 through p13, a combination of tower- and cell-based quantities were calculated (all variables are retained for backwards compatibility):

```

///////////////////////////////
// p11 new variable set           //
// MET for CAL + ICD Towers
float _SETT;
float _METTx;
float _METTy;
float _METT;

// MET for CAL + ICD Towers + Muon correction
float _SETTM;
float _METTMx;

```

```
float _METTMy;
float _METTM;

// Vis. Et for CAL + ICD Towers above eta limit, above tower threshold
float _SETTAS;
float _VETTASx;
float _VETTASY;
float _VETTAS;

// Vis. Et for CAL + ICD Towers below eta limit, above tower threshold
float _SETTBS;
float _VETTBSx;
float _VETTBSy;
float _VETTBS;

// Vis. Et for CAL + ICD Towers above eta limit, below tower threshold
float _SETTAN;
float _VETTANx;
float _VETTANY;
float _VETTAN;

// Vis. Et for CAL + ICD Towers below eta limit, below tower threshold
float _SETTBN;
float _VETTBNx;
float _VETTBNy;
float _VETTBN;

// MET for CAL + ICD Cells
float _SETC;
float _METCx;
float _METCy;
float _METC;

// MET for CAL + ICD Cells + Muon correction
float _SETCM;
float _METCMx;
float _METCMy;
float _METCM;

// Vis. Et for CAL + ICD Cells above eta limit, above cell threshold
float _SETCAS;
float _VETCASx;
float _VETCASY;
float _VETCAS;

// Vis. Et for CAL + ICD Cells below eta limit, above cell threshold
```

```

float _SETCBS;
float _VETCBSx;
float _VETCBSy;
float _VETCBS;

// Vis. Et for CAL + ICD Cells above eta limit, below cell threshold
float _SETCAN;
float _VETCANx;
float _VETCANy;
float _VETCAN;

// Vis. Et for CAL + ICD Cells below eta limit, below cell threshold
float _SETCBN;
float _VETCBNx;
float _VETCBNy;
float _VETCBN;

// Vis. Et for ICD Cells
float _SETICD;
float _VETICDx;
float _VETICDy;
float _VETICD;

// Vis. Et for NADA Cells
float _SETNADA;
float _VETNADAx;
float _VETNADAy;
float _VETNADA;

// Tight Muons
float _VETMUONx;
float _VETMUONY;
float _VETMUONz;
float _VETMUON;

```

The corresponding accessor methods are

```

void getMETT(float &SETT, float &METTx, float &METTy, float &METT);
void getMETTM(float &SETTM, float &METTMx, float &METTMy, float &METTM);
void getVETTAS(float &SETTAS, float &VETTASx, float &VETTASy, float &VETTAS);
void getVETTBS(float &SETTBS, float &VETTBSx, float &VETTBSy, float &VETTBS);
void getVETTAN(float &SETTAN, float &VETTANx, float &VETTANY, float &VETTAN);
void getVETTBN(float &SETTBN, float &VETTBNx, float &VETTBNy, float &VETTBN);
void getMETC(float &SETC, float &METCx, float &METCy, float &METC);
void getMETCM(float &SETCM, float &METCMx, float &METCMy, float &METCM);
void getVETCAS(float &SETCAS, float &VETCASx, float &VETCASy, float &VETCAS);
void getVETCBS(float &SETCBS, float &VETCBSx, float &VETCBSy, float &VETCBS);

```

```

void getVETCAN(float &SETCAN, float &VETCANx, float &VETCANy, float &VETCAN);
void getVETCBN(float &SETCBN, float &VETCBNx, float &VETCBNy, float &VETCBN);
void getVETICD(float &SETICD, float &VETICDx, float &VETICDy, float &VETICD);
void getVETNADA(float &SETNADA, float &VETNADAx, float &VETNADAY, float &VETNADA);
void getVETMUON(float &VETMUONx, float &VETMUONY, float &VETMUONz, float &VETMUON);

```

5 Appendix B: MissingET.rcp

```

////////////////////////////// File: MissingEt.rcp //////////////////////////////
// File: MissingEt.rcp                                         //
//                                                               //
// Purpose: Missing transverse energy control file           //
//                                                               //
// Created: 29-JUN-1998 John Womersley                         //
// History: 02-FEB-2000 Laurent Duflot: read file generated from //
//          simpp make missing ET from calorimeter cells        //
//          13-AUG-2000 Alan L. Stone: added ET threshold and      //
//          Eta high and low limit controls                      //
//          24-AUG-2001 Alan L. Stone: added rcp switch for abs(ET^2) //
//          14-NOV-2001 Alan L. Stone: modified default ET and eta   //
//          limits                                              //
//          3-DEC-2001 Vishnu Zutshi and Joe Steele: added known x,y //
//          offsets                                             //
//          11-FEB-2002 Alan L. Stone: added rcps for eta Limit,     //
//          Tower & Cell energy threshold to be used with new       //
//          variables                                           //
//          20-FEB-2002 Alan L. Stone: removed rcp for abs(ET2) - it //
//          does nothing, and is confusing                      //
////////////////////////////// PackageName = "MissingETReco" //
string PackageName = "MissingETReco"
string data_type = "MC"
string algo_type = "calorimeter"

// if useMCvertex is true: use Monte Carlo vertex
// if useD0vertex is true: use reconstructed vertex
// if both true use Monte Carlo vertex if there is no reconstructed vertex
bool useMCvertex    = false
bool usedD0vertex   = true

int etaLimit = 32           // abs(ieta) limit (inclusive)
float towerThreshold = 0.2  // Tower energy (GeV) threshold (inclusive)
float cellThreshold = 0.1   // Cell energy (GeV) threshold (inclusive)

float towerETLimitNoEta = 0.2 // Tower ET Threshold (GeV)

```

```

float towerETLimitWithEta = 0.2 // Tower ET Threshold (GeV)

int towerEtaLowLimit = 1      // Low Eta Value Limit (inclusive)
int towerEtaHighLimit = 32    // High Eta Value Limit (inclusive)

float XYZoffsets = (0. 0. -999.9)
// X, Y, and Z for data taken after the October-November shutdown in 2001.
// (Z is determined from the reco vertex for each event.)
// For data taken before Nov. of 2001, this line should be:
// float XYZoffsets = (0.435 0.505 -999.9)

```

6 Appendix B: MetAnalyze.rcp

```

///////////////////////////////
//
// File:      MetAnalyze.rcp
//
// Purpose:   rcp for MetAnalyze package
//
// Created:   27-JAN-2002 Alan L. Stone
//
// History:
///////////////////////////////

string PackageName = "MetAnalyze"

// what to do?
bool doMissingET = true
bool doEvtInfo = false

// turn on/off debugging printout
bool debugPrintout = false

// file for histograms and ntuples
string hbk_file = "metanalyze.ntpl"
string nt_manager = "HBOOK"

// MissingET chunk selector
RCP MET_Algo = <missingET MissingET>

string MetOldBlock = "METMET" // used as old block name and prefix of variable
string MetBlock = "MET"       // used as new block name and prefix of variables
string MetOldRingPrefix ="METRING" // used as old prefix for met rings

```

References

- [1] For a description of the calorimeter layout, layers, and towers, see D0 Note 774, “Calorimeter Addressing - Version 1.1”, Jim Linnemann (11/7/88).
- [2] D0 Note 3895, “mETRx: A MissingET Revertexing Tool”, Joseph Steele and Robert Hirosky (8/14/2001).
- [3] D0 Note 4057, “Improvement of the NADA Algorithm: Hot Cell Killing in D0 Run II Data”, Gregorio Bernardi Sophie Trincaz-Duvold. (12/1/02) D0 Note 3687, “NADA: A New Event by Event Hot Cell Killer”, G. Bernardi, B. Olivier, B. Knuteson, M. Strovink. (9/21/1999)
- [4] D0 Note 4146, “Technical description of the T42 algorithm for the calorimeter noise suppression”, Jean-Roch Vlimant, Ursula Bassler, Gregorio Bernardi, Sophie Trincaz-Duvold. (5/12/2003)